Tervezési minták egy OO programozási nyelvben. MVC, mint modell-nézet vezérlő minta és néhány másik tervezési minta.

Kalakai Zsolt

DPR8EE

Programtervezési minták

Az objektumorientált (OO) programozásban a programtervezési minták olyan ismétlődő megoldások, amelyeket gyakran előforduló problémákra használunk a szoftverfejlesztés során. Ezek a minták nem konkrét kódrészletek, hanem iránymutatások arra, hogyan lehet hatékonyan megoldani bizonyos típusú problémákat.
Az egyik legismertebb architektúra-minta az **MVC (Model-View-Controller)**, amely szétválasztja az alkalmazás adatkezelését, megjelenítését és az ezek közötti logikát. Ezenkívül számos tervezési mintát használhatunk az objektumorientált rendszerek tervezésében, mint például az **Egység (Singleton)**, **Stratégia (Strategy)** vagy **Megfigyelő (Observer)** minta.

Az MVC minta

Az **MVC** (Model-View-Controller, azaz Modell-Nézet-Vezérlő) egy architektúra-minta, amely különválasztja az alkalmazás három fő összetevőjét:

1. **Modell (Model):**
   Az alkalmazás adatainak és üzleti logikájának a reprezentációja. A modell felelős az adatok kezeléséért, például az adatok tárolásáért, módosításáért, valamint az alkalmazás üzleti szabályainak érvényesítéséért.
   Példa: Egy banki alkalmazásban a Modell tárolja a számlaegyenleget és kezeli az egyenleg módosítását.
2. **Nézet (View):**
   A felhasználó számára megjelenített adatokat kezeli. A nézet a modell adatait kérdezi le, és vizualizálja őket. Fontos, hogy a nézet „után” működik, azaz nem tartalmaz semmilyen üzleti logikát.
   Példa: Egy grafikus felület, amelyen a számlaegyenleget jelenítjük meg.
3. **Vezérlő (Controller):**
   Az interakciók kezeléséért felelős réteg. A vezérlő fogadja a felhasználói bemeneteket, és ez alapján meghívja a megfelelő modell- vagy nézetműveleteket.
   Példa: Egy gombkattintás, amely elindítja az utalási műveletet.

Az MVC működése:

A felhasználó műveletet hajt végre a felhasználói felületen (pl. kattint egy gombra).
A Vezérlő kezeli a műveletet, és ennek hatására utasítja a Modellt, hogy frissítse az adatokat.
A Modell értesíti a Nézetet, hogy az adatok megváltoztak.
A Nézet frissíti a megjelenítést a Modell új állapotának megfelelően.
Ez a folyamat világosan elkülöníti az egyes részegységek szerepét, megkönnyítve ezzel az alkalmazás fejlesztését és karbantartását.

Az MVC előnyei:

Moduláris felépítés: A Modell, Nézet és Vezérlő elkülönítése lehetővé teszi a független fejlesztést és tesztelést.

Karbantarthatóság: A kód könnyebben átlátható és módosítható, mivel az egyes komponensek jól elkülönülnek.

Újrafelhasználhatóság: A Modellek és Nézetek újrahasznosíthatók más projektekben vagy alkalmazásokban.

Rugalmasság: A felhasználói felület megváltoztatható anélkül, hogy az üzleti logikát módosítani kellene.

Az MVC hátrányai:

Komplexitás: Az MVC megértése és implementálása kezdetben bonyolult lehet kisebb projektek esetén.

Több fájl: A külön komponensek miatt nagyobb lehet a kód mennyisége és a fájlok száma.

Tanulási görbe: Az MVC használata tapasztalatot és gondos tervezést igényel.

1. Egység (Singleton) minta

Leírás:

Az Egység minta biztosítja, hogy egy adott osztályból legfeljebb egy példány jöjjön létre, és globális hozzáférést biztosít ehhez az egyetlen példányhoz.

Példa:

Egy konfigurációs fájl kezelése, ahol csak egy példány szükséges, hogy mindenhol ugyanazokat a beállításokat használjuk.

Előnyök:

- Globális hozzáférés biztosítása.
- Erőforrások megosztása (például adatbáziskapcsolat).

Hátrányok:

- A kódot nehezebb tesztelni, mert nehéz az Egységet mockolni.

2. Stratégia (Strategy) minta

Leírás:

A Stratégia minta lehetővé teszi, hogy egy algoritmust futásidőben lehessen cserélni különböző implementációkra anélkül, hogy az algoritmus használójának tudnia kellene a részleteket.

Példa:

Egy fizetési rendszer, ahol különböző fizetési módok (hitelkártya, PayPal, kriptovaluta) választhatók.

Előnyök:

- A kód újrahasznosíthatósága nő.
- Egyszerűbbé teszi az algoritmusok közötti váltást.

Hátrányok:

- Bonyolultabb kódszerkezet, ha túl sok stratégia van.

3. Megfigyelő (Observer) minta

A Megfigyelő minta lehetővé teszi, hogy egy objektum (alany) értesítse a hozzá kapcsolt megfigyelőket az állapotváltozásairól.

Példa:

Egy híroldal értesítései, ahol a felhasználók értesülnek az új cikkekről.

Előnyök:

- Egyszerűbbé teszi az események kezelését.
- Könnyen bővíthető, új megfigyelők hozzáadása nem igényli az alany módosítását.

Hátrányok:

- Az értesítési rendszer komplexebbé válhat nagyszámú megfigyelő esetén.

A MVVM (Model-View-ViewModel) tervezési minta

A MVVM (Model-View-ViewModel) tervezési minta egy szoftverarchitektúra-minta, amelyet gyakran használnak modern asztali, mobil és webes alkalmazások fejlesztésénél. Célja a kód újrafelhasználhatóságának növelése, az egységtesztelés megkönnyítése, valamint az alkalmazás különböző rétegeinek (adatkezelés, felhasználói felület, üzleti logika) szigorúbb elkülönítése.

A minta három fő komponense:

1. Model: A Model az alkalmazás adatkezeléséért és az üzleti logika megvalósításáért felelős réteg. Ez tartalmazza az adatokat és az azokkal kapcsolatos műveleteket, például az adatbázis-hozzáférést vagy a külső API-k hívását. A Model tisztán logikai elemekből áll, és nem tartalmaz felhasználói felületre vonatkozó kódot.

2. View: A View a felhasználói felület, amely felelős az adatok megjelenítéséért és a felhasználóval való interakció kezeléséért. A View-t úgy tervezik, hogy a lehető legkevesebb logikai kódot tartalmazza; fő feladata a ViewModel adatainak vizuális megjelenítése. Példák: HTML oldalak, grafikus felhasználói felületek vagy mobilalkalmazások felületei.

3. ViewModel: A ViewModel egy köztes réteg, amely a Model és a View között helyezkedik el. Ez felelős a Modelből származó adatok feldolgozásáért, valamint az adatok View számára történő átadásáért. A ViewModel tartalmazhat üzleti logikát is, amely közvetlenül a felhasználói élményhez kapcsolódik. Ezenkívül biztosítja a kétirányú adatkapcsolatot (data binding) a View és a ViewModel között.